Mathematical modeling

Математическое моделирование

UDC 004.021: 65.012.26
https://doi.org/10.32362/2500-316X-2023-11-1-60-69

RESEARCH ARTICLE

Algorithm for finding subcritical paths on network diagrams

Mikhail A. Anfyorov @

MIREA – Russian Technological University, Moscow, 119454 Russia
@ *Corresponding author, e-mail: anfyorov@inbox.ru*

Abstract

Objectives. Network diagrams are used as an information support element in planning and project management processes for structuring planned work and calculating project efficiency characteristics. In order to optimize and balance resources used in projects, it becomes necessary to locate in these models not only the critical path of the maximum weighted length, but also the subcritical paths closest to it having a shorter length in relation to it. The aim of the work is to synthesize and analyze an algorithm for finding *k*-shortest paths between the input and output network vertices, on which basis the above-mentioned subcritical paths can be identified.

Methods. The provisions of graph theory and group theory, as well as the method of dynamic programming, were used.

Results. An algorithm for finding *k*-shortest paths in contourless directed graphs having a strict order relation was developed. Abstract elements were defined according to group theory in graphs as *p*-contours, between which a multilevel structure of relations for implementing the necessary search of paths was then established. For substantiating the efficiency of the constructed algorithm, the validity of the main provisions was demonstrated as follows: firstly, the multilevel system of relations is exhaustive; secondly, there is no loss in the final solution during the operation of the algorithm; thirdly, the paths obtained as a result of the work of the algorithm satisfy the main required relation between them. Numerically, the algorithm was implemented by the dynamic programming method extended by means of an additional functional relationship, implying the presence of suboptimal policies.

Conclusions. The conducted runs of computational experiments confirmed the operability and efficiency of the software-implemented algorithm. The performed analysis demonstrated the good convergence characteristics of the proposed algorithm as compared with other algorithms of this class applied to network diagrams. On this basis, it can be recommended for practical use in project management information systems.

Keywords: project management, network diagram, critical path, algorithm, computational experiment

• **Submitted:** 27.03.2022 • **Revised:** 06.06.2022 • **Accepted:** 24.10.2022

For citation: Anfyorov M.A. Algorithm for finding subcritical paths on network diagrams. *Russ. Technol. J.* 2023;11(1):60–69. https://doi.org/10.32362/2500-316X-2023-11-1-60-69

Financial disclosure: The author has no a financial or property interest in any material or method mentioned.

The author declares no conflicts of interest.

НАУЧНАЯ СТАТЬЯ

Алгоритм поиска подкритических путей на сетевых графиках

М.А. Анфёров [@]

МИРЭА – Российский технологический университет, Москва, 119454 Россия
[@] *Автор для переписки, e-mail: anfyorov@inbox.ru*

Резюме

Цели. Информационная поддержка процессов планирования и управления проектами использует в качестве модели сетевые графики, помогающие в формировании структуры планируемых работ и расчете характеристик эффективности проекта. С целью оптимизации и выравнивания ресурсов, используемых в проектах, возникает необходимость нахождения на этих моделях не только критического пути максимальной взвешенной длины, но и ближайших к нему подкритических путей с меньшей по отношению к нему длиной. Цель работы – синтез и анализ алгоритма поиска *k*-кратчайших путей между вершинами входа и выхода сети, позволяющего идентифицировать вышеназванные подкритические пути.

Методы. Использованы положения теории графов и теории групп, а также метод динамического программирования.

Результаты. Разработан алгоритм поиска *k*-кратчайших путей на ориентированных графах без контуров с отношением строгого порядка. С использованием теории групп на графах были определены абстрактные элементы – *p*-контуры, между которыми была установлена многоуровневая структура отношений, позволившая реализовывать необходимый поиск путей. В рамках обоснования работоспособности построенного алгоритма доказана справедливость основных положений: во-первых, многоуровневая система отношений является исчерпывающей; во-вторых, не происходит потерь в окончательном решении в процессе работы алгоритма; в-третьих, пути, найденные в результате работы алгоритма, удовлетворяют основному требуемому соотношению между ними. Численно алгоритм реализован методом динамического программирования, который был расширен за счет использования дополнительного функционального соотношения, предполагающего наличие подоптимальных политик.

Выводы. Проведенная серия вычислительных экспериментов подтвердила работоспособность и эффективность программно реализованного алгоритма. Выполненный анализ показал хорошие характеристики сходимости предложенного алгоритма в сравнении с алгоритмами данного класса, применяемыми к сетевым графикам. Это позволяет рекомендовать его к практическому использованию в информационных системах управления проектами.

Ключевые слова: управление проектами, сетевой график, критический путь, алгоритм, вычислительный эксперимент

• **Поступила:** 27.03.2022 • **Доработана:** 06.06.2022 • **Принята к опубликованию:** 24.10.2022

Для цитирования: Анфёров М.А. Алгоритм поиска подкритических путей на сетевых графиках. *Russ. Technol. J.* 2023;11(1):60–69. https://doi.org/10.32362/2500-316X-2023-11-1-60-69

Прозрачность финансовой деятельности: Автор не имеет финансовой заинтересованности в представленных материалах или методах.

Автор заявляет об отсутствии конфликта интересов.

INTRODUCTION

Established project management methodologies use a contourless network-oriented graph as the main model for displaying the structure of the mutual dependence of project work stages and calculating the time characteristics of these stages, as well as those of the project as a whole [1–3]. In the terminology of network planning and management, such a model is referred to as a network diagram, which may be constructed according to one of two principles. In the first case, typical for the Project Evaluation and Review Technique (PERT) methodology, the work in the model display uses graph arcs to represent the "event-work" principle, while the second uses graph vertices according to the "work-relationship" principle. However, both of these approaches involve the definition of

a critical path on the network diagram, having a maximum weighted length and connecting a hanging vertex to a dead end, which may correspond to the beginning or end of the project. In this case, the length is calculated based on the value of the critical work execution time displayed by the active elements of the path (arcs or vertices). The use of a critical path is an integral part of network planning and project management methodology, as reflected in studies at the stage of its theoretical development (an extensive bibliography of this period is presented in [4]) and subsequent development [5–8].

Actual projects are implemented in various areas under conditions of limited material, labor and financial resources. This implies the development of project management theory in the direction of optimizing the structure of the project according to the criterion of its minimum cost under particular resource constraints [7–12]. In this case, the question arises of the need to search the network diagram not only for critical, but also for subcritical paths in order to implement the above-mentioned optimization and leveling of resources. If the critical path Ω_0 can be defined as

$$\Omega_0 : L(\Omega_0) = \max_{\Omega_j \in \mathbf{\Omega}} \{L(\Omega_j)\}, \quad (1)$$

where $L(\Omega_0)$ is the length of the critical path, and $\mathbf{\Omega}$ is the set of all full paths on the network diagram, then the subcritical paths will form an ordered set of full paths of the graph $\{\Omega_1, \Omega_2, ...\}$, characterized as follows:

$$L(\Omega_0) \geq L(\Omega_1); L(\Omega_j) \geq L(\Omega_{j+1}). \quad (2)$$

To find the critical path, it is sufficient to use any algorithm to find the shortest weighted path between the vertices that represent the beginning and end points of the project on the network diagram after changing the weights of its active elements (time to complete the work) to negative. Such algorithms can be based on dynamic programming [13, 14] or heuristic methods [15]. When obtaining subcritical paths, algorithms for finding *k*-shortest paths between graph vertices either use dynamic programming directly [16–22] or with repeated application of the shortest path search [23–25], or other contemporary approaches [26–28], as well as those focused on more complex network designs [29–32].

The present paper describes an algorithm for finding *k*-shortest paths on network graphs based on the selection of elements of a higher order and structuring relations between them.

STRUCTURING RELATIONSHIPS ON GRAPHS

The performed studies are focused on the use of network graphs of the "works-connections" type, using as a model network directed graphs **G(X, U)** (**X** is a set of vertices, **U** is a set of arcs) without contours with a strict order relation, and with division into layers [33] (Fig. 1).

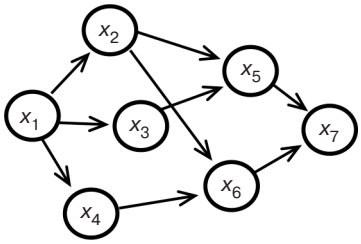

Fig. 1. Network diagram

Let us construct a free Abelian group **P(U)** over the generating set **U** of all arcs of the graph **G = (X, U)** as follows. As elements of **P(U)** we will consider the set of formal linear combinations of elements from **U** with integer coefficients in the form of

$$p_j = \sum_{i=1}^{n} (\gamma_i \cdot u_i), \; p_j \in \mathbf{P(U)}; \; u_i \in \mathbf{U}; \gamma_i = 0, \pm 1, ...,$$

where *n* is the number of graph arcs.

As a binary additive operation, we define the sum of elements from the set **P(U)** by the formula:

$$p_j = \sum_{i=1}^{n} (\gamma_i \cdot u_i) + \sum_{i=1}^{n} (\gamma'_i \cdot u_i) = \sum_{i=1}^{n} (\gamma_i + \gamma'_i) \cdot u_i.$$

Similarly, we construct a group **H(X)** over the set **X** of all vertices of the graph, defining its elements as $h_j = \sum_{i=1}^{m} (\gamma_i \cdot x_i), \; h_j \in \mathbf{H(X)}; \; x_i \in \mathbf{X}; \gamma_i = 0, \pm 1, ...,$ where *m* is the number of vertices. In what follows, when writing elements of the groups **P(U)** and **H(X)**, we will omit their constituent elements that have zero coefficients.

Definition 1. A differential *d* of a group **P(U)** is called a homomorphism *d*: **P(U)** → **H(X)** defined as follows:

1) if $u_t = (x_i, x_j)$, then $du_t = x_j - x_i$;
2) if $p_q = \Sigma\gamma_t u_t$, then $dp_q = \Sigma\gamma_t du_t$

for $p_q \in \mathbf{P(U)}$; $u_t \in \mathbf{U}$; $x_i, x_j \in \mathbf{X}$.

Definition 2. An element $r \in \mathbf{P(U)}$ is called a *p*-contour if $dr = 0$. The subgroup $\mathbf{R} = Ker\, d \subset \mathbf{P(U)}$ is called a subgroup of *p*-contours.

Lemma 1. The sum of several *p*-contours is a *p*-contour.

Proof. First, the sum Σr_i is an element of the group **P(U)** as a result of addition of its elements. Second, the distributivity of the mapping *d* according to *Definition 1* allows us to write down $d\Sigma r_i = \Sigma dr_i = 0$. The assertion of the lemma is proved.

Using the generally accepted concept of a path on a graph as a connected finite sequence of arcs, we will denote it as

$$\Omega = \{u_1, u_2, \ldots, u_w\}. \quad (3)$$

Moreover, if $u = (x_1, x_2)$, то $x_1 = u^-$, $x_2 = u^+$. We will determine the length of this path through the above numerical weighted estimates of the vertices $\varepsilon(x)$ by the expression

$$L(\Omega) = \varepsilon(u_1^-) + \sum_{i=1}^{w} \varepsilon(u_1^+). \quad (4)$$

If the path connects the input vertex x' with the output vertex x'' of the network, then we will call such a path complete.

We define the mapping φ as follows:

$$\varphi(\{\gamma_1 u_1, \gamma_2 u_2, ..., \gamma_n u_n\}) = \sum_{i=1}^{n} \gamma_i u_i, \gamma_i > 0, \quad (5)$$

and also, the converse to the above:

$$\varphi'\left(\sum_{i=1}^{n} \gamma_i u_i\right) = \{|\gamma_1| u_1, |\gamma_2| u_2, ..., |\gamma_n| u_n\}. \quad (6)$$

Definition 3. The image of a path $\Omega \subset \mathbf{U}$ on the graph $\mathbf{G} = (\mathbf{X}, \mathbf{U})$ is called a mapping $\varphi(\Omega)$.

Lemma 2. The differential of the image of any complete path on a network graph is defined as $d\varphi(\Omega) = x'' - x'$.

Proof. Since the full path Ω on the graphs under consideration is a simple path that does not have multiple arcs, then its image, taking into account (3), can be written as $\varphi(\Omega) = u_1 + u_2 + \ldots + u_w$. In accordance with *Definition 1*, we obtain either $d\varphi(\Omega) = du_1 + du_2 + \ldots + du_w$ or $d\varphi(\Omega) = (u_1^+ - u_1^-) + (u_2^+ - u_2^-) + ... + (u_w^+ - u_w^-)$, or

$$d\varphi(\Omega) = -u_1^- + (u_1^+ - u_2^-) + ... + (u_{w-1}^+ - u_w^-) + u_w^+. \quad (7)$$

The incidence property of path arcs implies $u_i^+ = u_{i+1}^-, i \in [1, w-1]$. Therefore, expression (7) can be written in the form $d\varphi(\Omega) = u_w^+ - u_1^-$ or $d\varphi(\Omega) = x'' - x'$.

Lemma 3. The difference between the images of two complete paths on a network graph is a *p*-contour.

Proof. The difference between the images of the full paths $\varphi(\Omega_i)$ and $\varphi(\Omega_j)$, being the result of the additive function of adding two elements of the group $\mathbf{P}(\mathbf{U})$, also belongs to this group. On the other hand, the differential of this difference is defined as $d[\varphi(\Omega_i) - \varphi(\Omega_j)] = d\varphi(\Omega_i) - d\varphi(\Omega_j)$. However, by the assertion of *Lemma 2* $d\varphi(\Omega_i) = d\varphi(\Omega_j)$. Hence, $d[\varphi(\Omega_i) - \varphi(\Omega_j)] = 0$, which proves the lemma.

Let us define the mapping α as follows:

$$\alpha(p) = \sum_{u_i \in p}^{n} \gamma_i \cdot \varepsilon(u_i^+). \quad (8)$$

Definition 4. The value of the *p*-contour *r* is the value $\alpha(r)$.

Lemma 4. The difference between the lengths of two complete paths on a network graph is equal to the value *p* of the contour formed by the difference of their images.

Proof. Let $\Omega_i = \{u_{i1}, u_{i2}, \ldots, u_{iw}\}$ and $\Omega_j = \{u_{j1}, u_{j2}, \ldots, u_{js}\}$ be any two complete paths of the graph.

Then their lengths, in accordance with (4), can be written as $L(\Omega_i) = \varepsilon(x') + \sum_{t=1}^{w} \varepsilon(u_{it}^+)$, $L(\Omega_j) = \varepsilon(x') + \sum_{t=1}^{s} \varepsilon(u_{jt}^+)$, and the difference in lengths as

$$L(\Omega_i) - L(\Omega_j) = \sum_{t=1}^{w} \varepsilon(u_{it}^+) - \sum_{t=1}^{s} \varepsilon(u_{jt}^+). \quad (9)$$

On the other hand, in accordance with *Lemma 3*, the *p*-contour defined by the images of these full paths can be written, taking into account expression (5), as $r = \varphi(\Omega_i) - \varphi(\Omega_j) = u_{i1} + u_{i2} + ... + u_{iw} - u_{j1} - u_{j2} - ... - u_{js}$, and the value of this *p*-contour, taking into account (8), as

$$\alpha(r) = \sum_{t=1}^{w} \varepsilon(u_{it}^+) - \sum_{t=1}^{s} \varepsilon(u_{jt}^+). \quad (10)$$

Comparing expressions (9) and (10), we conclude that the lemma is true.

Definition 5. An elementary *p*-contour $g^{(ab)}$ with respect to an arc of a graph $u = (a, b) \in \mathbf{U}$ is an element of the group $\mathbf{P}(\mathbf{U})$, defined as the sum of this arc with the difference between the images of the shortest weighted paths connecting the vertices x' and a, as well as the vertices x' and b (see Fig. 2). In this case, the elementary *p*-contours are oriented to the shortest paths, since the main goal is to find *k*-shortest paths of the graph.

Definition 6. An arc u_i is said to be incident to the path Ω_0 at the vertex *b* if the following conditions are met:

1) $u_i \notin \Omega_0$;
2) $\exists u_i \in \Omega_0, u_i^+ = u_j^+ = b$.

The vertex *b* is called the vertex of the section of the path Ω_0 (Fig. 2).

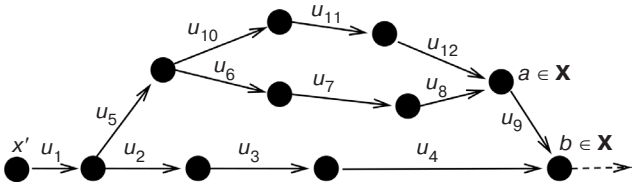

Fig. 2. $\{u_1, u_5, u_6, u_7, u_8\}$ – shortest path between vertices x' and a. $\{u_1, u_2, u_3, u_8\}$ – shortest path between vertices x' and b. $\{u_5 + u_6 + u_7 + u_8 + u_9 - u_2 - u_3 - u_4\}$ – elementary p-contour ($g^{(ab)}$), based along the path $\{u_1, u_2, u_3, u_4, ...\}$ along the incident arc u_9 at the vertex of the section b: $g^{(ab)} \equiv r^{bi}$

Definition 7. An elementary p-contour with respect to an arc incident to the path Ω at the vertex of the section $b \in \mathbf{X}$ is called based on this path and is denoted as r^b (Fig. 2)[1].

In connection with the consideration of the full paths of the graph in the framework of the network planning problem, the p-contour based on the path Ω_0, which allows us to determine another full path Ω_s, will be called generating. Based on lemmas 3 and 4, we write the relations

$$\varphi(\Omega') = \varphi(\Omega_0) + r^b,\ L(\Omega') = L(\Omega_0) + \alpha(r^b), \quad (11)$$

which are further used in the algorithm for finding k-shortest paths of the graph.

In order to use the models to efficiently solve numerical problems, an ordered structure of relations between the complete paths of the network graph can be constructed on the basis of the introduced mathematical objects.

With regard to the development of the described algorithm, we consider the problem of finding an ordered set of full paths

$$\{\Omega_0, \Omega_1, \ldots, \Omega_k\},\ L(\Omega_i) \le L(\Omega_{i+1}),\ i \le [0, k-1]. \quad (12)$$

The system of relationships between paths is built relative to the full path Ω_0, which has a minimum length $\Omega_0 : L(\Omega_0) = \min\limits_{\Omega_j \in \mathbf{\Omega}} \{L(\Omega_j)\}$, can be easily obtained using one of the well-known algorithms, for example as given in [13, 14]. This system is described by a hierarchical multilevel structure of shortest path Ω_0 generating p-contours in the form of a graph $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$, $\mathbf{V} = \mathbf{R}_0 \times \mathbf{R}_0$. Here $\mathbf{R}_0$ is the complete set of generating p-contours, while $\mathbf{V}$ is the set of relations between them that connect the generating p-contours of adjacent levels. So, for the upper 0th level, the connection of p-contours[2] $\mathbf{R}_{00} = \{r_0^b\}$ with p-contours of the next 1st level $\mathbf{R}_{01}$ is represented by the relation

$$r_1^{be} = r_0^b + r^e;\ \alpha(r_1^{be}) = \alpha(r_0^b) + \alpha(r^e), \quad (13)$$

where b is the vertex of the section of the path Ω_0; e is the vertex of the section of the path formed by the p-contour r_0^b (11) located on this path to the left of the vertex b (i.e., $e < b$)[3]. The grouping by levels defines the entire set of generating p-contours $\mathbf{R}_0 = \bigcup\limits_Z \mathbf{R}_{0Z}$.

DESCRIPTION OF THE ALGORITHM

The algorithm is presented below in a less compact form, excluding loops, in order to visually show the finiteness of the number of steps performed.

Step 1. Find the full path Ω_0 with the minimum length.

Step 2. Put the set $\mathbf{R}_{00} = \{r_1, r_2, r_3, ...\}$ of generating p-contours of the 0th level in ascending order of their values[4] and exclude from further consideration the p-contours that are below the kth place in the resulting sequence; $\bar{\mathbf{R}} = \mathbf{R}_{00}$. Find the full path Ω_1 through relations (11) using the p-contour r_1, which is the first in the sequence.

Step 3. Include in the set $\bar{\mathbf{R}}$ the p-contours of the next level, determined by the p-contour r_1 through relations (13) over all vertices of the section, excluding the p-contour r_1 itself. Order the set $\bar{\mathbf{R}}$ according to the relation $\alpha(r_i) \le \alpha(r_{i+1})$ and exclude from further consideration the p-contours that are below $(k - 1)$ places in the resulting sequence.

Find the full path Ω_2 through relations (11) using the p-contour r_1.

. . . . . .

Step(s). Include in the set $\bar{\mathbf{R}}$ p-contours of the next level, determined by the p-contour r_1 through relations (13) over all vertices of the section, excluding the p-contour r_1 itself. Order the set $\bar{\mathbf{R}}$ according to the relation $\alpha(r_i) \le \alpha(r_{i+1})$ and exclude from further consideration p-contours that are in the resulting sequence below $[k - (s - 2)]$ place[5].

[1] Several elementary contours can be based on the vertex b in the presence of several incident arcs.

[2] An additional subscript is introduced to denote the level number in the system of generating p-contours.

[3] This condition is related to the orientation to the input vertex x' when constructing elementary p-contours (see *Definition 5*).

[4] A simplified designation of generating contours was introduced in order to better understand the operation of the algorithm. The subscript shows the place of the contour in the ordered set.

[5] The particular case is when the number of elements of the set $\mathbf{R}_0$ is less than $[k - (i - 2)]$. In this case, contours are not excluded from the set, which does not affect the course of all reasoning and the final result.

Find the full path Ω_{s-1} through relations (11) using the p-contour r_1.

. . . . . .

Step (k + 1). Include in the set $\bar{\mathbf{R}}$ p-contours of the next level, which are determined by the p-contour r_1 through relations (13) over all vertices of the section, excluding the p-contour r_1 itself. Order the set $\bar{\mathbf{R}}$ according to the relation $\alpha(r_i) \le \alpha(r_{i+1})$ and exclude from further consideration the p-contours that are below $[k - (k - 1)] = 1$ place in the resulting sequence.

Find the full path Ω_k through relations (11) using the p-contour r_1.

The implementation of the algorithm is based on the search for generating p-contours and the full path Ω_0, which has a minimum length. This problem is effectively solved by the dynamic programming method. The main Bellman recursive functional relation for the problem of finding the shortest path[6] $\Omega^{1t}_{\min}$ connecting the input vertex (x_1) with any vertex x_t can be written as

$$L(\Omega^{1t}_{\min}) = \min_i \{\varepsilon(x_t) + L(\Omega^{1i}_{\min})\}, i \in \mathbf{I}_t, \quad (14)$$

where $\mathbf{I}_t$ is a subset of graph vertex numbers defined by the condition $\forall x_i\ (i \in \mathbf{I}_t)\ \exists u_y$ such that $x_i = u_y^-, x_t = u_y^+$. An element from $\mathbf{I}_t$, which is the optimal policy defined by (14), will be denoted as i_o.

To search for generating p-contours based on the vertex of the section t, one more functional relation must be defined in the form

$$L(\Omega^{1t}_j) = \varepsilon(x_t) + L(\Omega^{1j}_{\min}),\ j \in \mathbf{J}_t, \mathbf{J}_t = \mathbf{I}_t / i_o, \quad (15)$$

where Ω^{1t}_j is the path connecting vertices x_1 and x_t and passing through vertex x_j (Fig. 3).

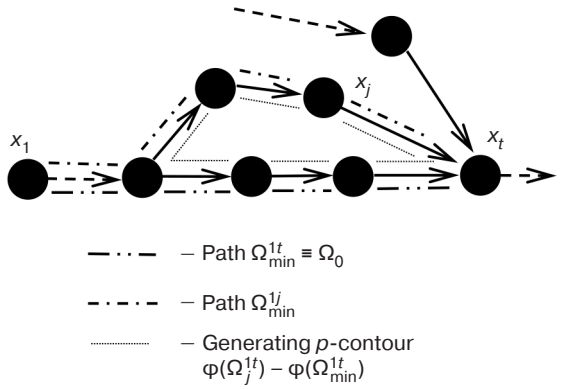

Fig. 3. Graphical explanation for the above calculations

Relation (15) implies the presence of suboptimal policies in the dynamic programming method, whose presence demonstrated by R. Bellman [34]. This relation defines the set of incident arcs $\{(x_j, x_t)\}, j \in \mathbf{J}_t\ \{(x_j, x_t)\}$ at the vertex of the section x_t and the corresponding generating p-contours through the knowledge of paths $\Omega^{1j}_{\min}$ (see *Definition 5*).

A software implementation of the algorithm in the Delphi programming language was performed. The conducted testing of the program confirmed the efficiency and effectiveness of the presented algorithm.

JUSTIFICATION OF THE ALGORITHM

To prove the efficiency of the constructed algorithm, it is necessary to verify the validity of the following provisions:

- firstly, any generating p-contour of the complete path Ω_0 is described by the system $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$;
- secondly, there is no loss in the final solution during the operation of the algorithm;
- thirdly, the paths found as a result of the algorithm's operation meet the condition (12).

The validity of the first proposition is confirmed by the following theorem.

Theorem 1. Any generating p-contour of the complete path Ω_0 is described by the system $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$.

Proof. Let r_x be an arbitrarily chosen generating p-contour of the path Ω_0. Let us show that this p-contour is described by one of the levels of the system $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$.

There necessarily exists an arc u_0 incident to the path Ω_0 with the section vertex b_0 such that $u_0 \in \varphi'(r_x)$. This arc corresponds to an elementary p-contour $g^{u_0} = r^{b_0}$ based on Ω_0.

If $r^{b_0} = r_x$, then the p-contour r_x is described by the system $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$ at the 0th level. If the equality is not maintained, then $\alpha(r^{b_0}) \le \alpha(r_x)$, which follows from *Definition 5*; therefore, the inequality $L(\Omega_1) \le L(\Omega_x)$ is true, where Ω_1 and Ω_x are full paths defined by p-contours r^{b_0} and r_x through relations (11), i.e.

$$\varphi(\Omega_1) = \varphi(\Omega_0) + r^{b_0},\ L(\Omega_1) = L(\Omega_0) + \alpha(r^{b_0}), \quad (16)$$

$$\varphi(\Omega_x) = \varphi(\Omega_0) + r_x,\ L(\Omega_x) = L(\Omega_0) + \alpha(r_x). \quad (17)$$

Subtracting equalities (16) and (17) term by term, and having transformed the result, we obtain $\varphi(\Omega_x) = \varphi(\Omega_1) + (r_x - r^{b_0})$, where $(r_x - r^{b_0})$ is the generating p-contour of the path Ω_1. Moreover, its value (obtained by the same subtraction of equalities (16) and (17)) is equal to $\alpha(r_x) - \alpha(r^{b_0}) \ge 0$.

Since $u_0 \in \Omega_1$, then the vertex of the section b_1 of the p-contour $(r_x - r^{b_0})$ is to the left of b_0.

Let us go to the next similar step of calculations. There is an arc u_1 incident to the path Ω_1 with the section

[6] The superscripts show the numbers of connected vertices.

vertex b_1 such that $u_1 \in \varphi'(r_x - r^{b_0})$. This arc corresponds to an elementary p-contour $g^{u_1} = r^{b_1}$ based on Ω_0.

If $r^{b_1} = r_x - r^{b_0}$, then p-contour $r_x = r^{b_0} + r^{b_1}$, i.e. is described by the system $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$ at the 1st level. If the equality is not maintained, then $\alpha(r^{b_1}) \le \alpha(r_x - r^{b_0})$ or $\alpha(r_x) \ge \alpha(r^{b_0}) + \alpha(r^{b_1})$, and, hence, $L(\Omega_2) \le L(\Omega_x)$, where Ω_2 is the full path determined by the p-contour r^{b_1} through relations (11) as $\varphi(\Omega_2) = \varphi(\Omega_1) + r^{b_1}$ or taking into account (16)

$$\varphi(\Omega_2) = \varphi(\Omega_0) + r^{b_0} + r^{b_1},\ L(\Omega_2) =$$

$$= L(\Omega_0) + \alpha(r^{b_0} + r^{b_1}). \qquad (18)$$

As a result, by analogy, there is a p-contour of the path Ω_2 generating p in the form

$$(r_x - r^{b_0} - r^{b_1}),\ \alpha(r_x - r^{b_0}) - \alpha(r^{b_1}) \ge 0.$$

Thus, at any Nth step calculations, the non-fulfillment of the equality condition

$$r^{b_N} = r_x - r^{b_0} - r^{b_1} - \ldots - r^{b_{N-1}} \qquad (19)$$

leads to the next step. However, since the number of steps is limited, therefore, at a certain step, equality (19) will be maintained (i.e., r^{b_N} will be an elementary p-contour). This means that the p-contour r_x of the path Ω_0 will be described by the system $\mathbf{G} = (\mathbf{R}_0, \mathbf{V})$ at the $(N-1)$ level.

The limited number of steps is confirmed by the following considerations. Firstly, the incident arc chosen at any step belongs to the path Ω_x; secondly, as noted above, the section vertex is located to the left of that chosen at the previous step. Since the number of arcs of the path Ω_x is limited, the number of steps in the main reasoning is also limited.

Since the p-contour r_x was chosen arbitrarily, the theorem can be considered to have been proved.

To validate the second statement, we introduce into consideration the vector $\mathbf{Q}_j = [\alpha(r_{1j}), \alpha(r_{2j}), \ldots, \alpha(r_{(k+1-j)j})]$, whose elements are the values of p-contours that make up the set $\bar{\mathbf{R}}$ when the algorithm works at the stage of determining the path Ω_j (the number of the path determines the second subscript in the designation of these p-contours). Recall that the elements of the vector $\mathbf{Q}_j$ form a non-decreasing sequence.

We also introduce a vector $\mathbf{W} = [\alpha(r_{k1}), \alpha(r_{(k-1)2}), \ldots, \alpha(r_{1k})]$ (here, the semantic meaning of the indices corresponds to the vector $\mathbf{Q}_j$), whose elements $\alpha(r_{(k+1-j)j})$ are the values of p-contours that are at the last place in the set $\bar{\mathbf{R}}$ at the stage of determining the path Ω_j.

Theorem 2. The elements of the vector $\mathbf{W}$ form a non-increasing sequence.

Proof. Consider an arbitrarily chosen jth element of the vector $\mathbf{W}$, i.e., $\alpha(r_{(k+1-j)j})$, which is also the last element of the vector $\mathbf{Q}_j$. We are interested in the process of transition during the operation of the algorithm from the vector $\mathbf{Q}_j$ to the vector $\mathbf{Q}_{j+1}$ looking to the appearance of a new element $\alpha(r_{(k-j)(j+1)})$ of the vector $\mathbf{W}$. It consists of the following stages:

- the first element of the vector $\mathbf{Q}_j$ is excluded;
- new elements are added to the remaining elements, corresponding to the lower level p-contours included in the consideration;
- ordering of the resulting set;
- exclusion from further consideration of extra p-contours.

Exclusion of the first element of the vector $\mathbf{Q}_j$ cannot affect the choice of the element $\alpha(r_{(k-j)(j+1)})$.

The result of the procedures that should follow will not change if they are performed in a slightly different order. To the remaining elements of the vector $\mathbf{Q}_j$ (their number now corresponds to the required number of elements of the vector $\mathbf{Q}_{j+1}$), we will add one element of the new set of p-contour values. After each such addition, we will arrange the set $\bar{\mathbf{R}}$ and eliminate the extra element.

The elements of the vector $\mathbf{Q}_{j+1}$ before adding new elements are determined as follows: $\alpha(r_{1(j+1)}) = \alpha(r_{2j})$, $\alpha(r_{2(j+1)}) = \alpha(r_{3j})$, ..., $\alpha(r_{(k-j)(j+1)}) = \alpha(r_{(k+1-j)j})$. For the added element $\alpha(r_y)$, there are two possible cases: either $\alpha(r_y) \ge \alpha(r_{(k+1-j)j})$, or $\alpha(r_{(t+1)j}) \ge \alpha(r_y) \ge \alpha(r_{tj})$, $t \in [2, k-j]$. In the first case, it is the element $\alpha(r_y)$ that is excluded from further consideration, which will not change the ordered position of the remaining elements of the vector $\mathbf{Q}_j$. In the second case, following ordering, the element $\alpha(r_y)$ will take the place of the element $\alpha(r_{(t+1)j})$, the element $\alpha(r_{(t+1)j})$ will replace element $\alpha(r_{(t+2)j})$, etc.

Ultimately, the element $\alpha(r_{(k-j)j})$ will take the place of the element $\alpha(r_{(k+1-j)j})$ that will be removed. But, since $\alpha(r_{(k-j)j}) \le \alpha(r_{(k+1-j)j})$ (due to the definition of the vector $\mathbf{Q}_j$), then in this case there will be no increase in the value of the last element either, i.e., the element $\alpha(r_{(k-j)(j+1)})$ cannot be greater than the element $\alpha(r_{(k+1-j)j})$.

Since an arbitrary element of the vector $\mathbf{W}$ was chosen as the jth element, *the theorem can be considered to have been proved.*

Using *Theorem 2*, it is easy to prove that there are no losses in the final solution during the operation of the algorithm. Indeed, losses can occur only in the procedure of excluding p-contours from further consideration after they are ordered at each step[7]. However, at any step of

[7] The exclusion of the contour that is the first does not lead to losses, since on the one hand it is used to obtain a solution, and on the other hand, all lower-level contours generated by it fall into consideration.

the algorithm, the values of the eliminated p-contours are not less than the element $\alpha(r_{(k+1-j)j})$ of the vector $\mathbf{Q}_j$. Based on *Theorem 2*, we can conclude that these values are also certainly not less than the subsequent elements of the vector $\mathbf{W}$: $\alpha(r_{(k-j)(j+1)})$, $\alpha(r_{(k-j-1)(j+2)})$, ..., $\alpha(r_{1k})$, i.e., they should not be considered at the remaining steps of the algorithm and cannot be included in the final decision.

To validate the last statement, we introduce into consideration a vector consisting of the first elements of the vectors $\mathbf{Q}_j, j \in [1, k]$: $\bar{\mathbf{W}} = [\alpha(r_{11}), \alpha(r_{12}), ..., \alpha(r_{1k})]$, participating in the formation of the final solution through relations (11). Then the validity of condition (12) is affirmed by the following theorem.

Theorem 3. Vector elements $\bar{\mathbf{W}}$ form a non-decreasing sequence.

Proof. Consider an arbitrarily chosen jth element $\alpha(r_{1j})$ of the vector $\bar{\mathbf{W}}$. This element is also the first element of the vector $\mathbf{Q}_j$. In the process of transition from the vector $\mathbf{Q}_j$ to the vector $\mathbf{Q}_{j+1}$, the added new set of p-contours is characterized by their values certainly being not less than $\alpha(r_{1j})$; this is because they are at the next level after the p-contour and are determined by this p-contour through relations (13).

After ordering the set $\bar{\mathbf{R}}$, the element $\alpha(r_{1(j+1)})$ will be replaced by the element $\alpha(r_{2j})$ or by the smallest from the added set $\alpha_{\min}(r_{\text{add}})$, i.e., $\alpha(r_{1(j+1)}) = \min\{\alpha(r_{2j}), \alpha_{\min}(r_{\text{add}})\}$. But since $\alpha(r_{2j}) \geq \alpha(r_{1j})$, $\alpha_{\min}(r_{\text{add}}) \geq \alpha(r_{1j})$, then $\alpha(r_{1(j+1)}) \geq \alpha(r_{1j})$.

Since an arbitrary element of the vector $\bar{\mathbf{W}}$ was chosen as the jth one, the last inequality *proves the theorem.*

The performance of the algorithm was evaluated in comparison with the "double sweep" algorithm, according to which its author conducted serious computational experiments [35]. Moreover, the analytical performance assessment for algorithms of this category is far from real results, since the computation time strongly depends on the configuration of the networks used; moreover, the generalized operations performed during the execution of the algorithm cannot be unambiguously mapped into elementary operations of addition and comparison. Thus, for the mentioned algorithm, which belongs to the class of the most productive algorithms of this category, the analytical estimated computation time is of the order of $O(kN^3)$ [19, 20], while computational experiments show other results [35]; instead of a linear one, a polynomial dependence is observed of computation time τ on the number of shortest paths k:

$$\tau = 0.8457 + 0.1616\,k + 0.0260\,k^2.$$

At the same time, it should be noted that studies [35] were carried out not on network- but directed graphs, whose vertices formed a lattice structure with contours. The dimension of the graphs was taken into account through the dimensions and configuration of the lattice formed by the vertices. The weights of the arcs were generated by random integers in the range up to 100. In addition, the value of k was significantly limited.

Therefore, in connection with the search for subcritical paths on network graphs, a series of computational experiments was carried out on an Intel microprocessor with a clock frequency of 1.7 GHz in order to compare the algorithm obtained in this work with the Double sweep algorithm. The results of the dependence of the computation time (t) on the number of subcritical paths (k) and the dimensional characteristic of the network graphs (z) are shown in Fig. 4. The dimensional characteristic z is the product of the total number of graph arcs and the number of arcs in the path of maximum length connecting the input and output vertices of the network.

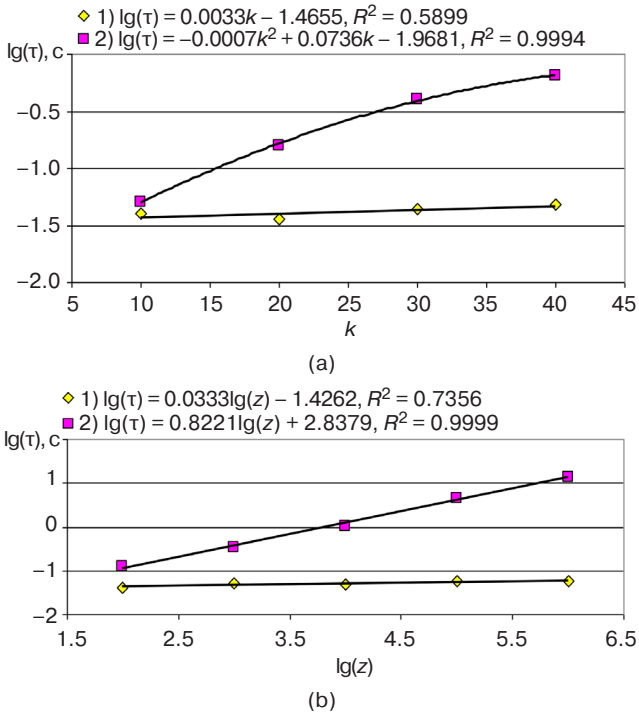

Fig. 4. Comparative analysis of algorithms:
(a) z = 1600; (b) k = 40.
1 – Algorithm used in this work; 2 – Double sweep algorithm

The higher performance of the algorithm presented in the paper is explained by its special orientation to the considered class of network graphs, while other algorithms, including the Double sweep one, are more universal in relation to computed graphs.

CONCLUSIONS

The presented algorithm implements the search for k-shortest paths without contourless directed graphs having a strict order relation used as models (so-called network graphs) in network planning and project management problems. This can be used to find

subcritical paths on these models in order to align and optimize the resources used in a project.

The above-described features of the graphs were instrumental in building a multilevel structure of relations between specially introduced abstractions (*p*-contours) that display the structural elements of graphs. This served as the basis for the development of the algorithm, which was numerically implemented by the dynamic programming method and extended through the use of an additional functional relation.

The narrow focus of the algorithm on graphs used in project management determined its high performance, which was confirmed by a series of comparative computational experiments. The efficiency of the algorithm is stable relative to the size of the computed graphs and the number of subcritical paths. For this reason, it can be recommended for practical use in project management information systems.

REFERENCES

1. Hagstrom J.N. Computing the probability distribution of project duration in a PERT network. *Networks*. 1990;20(2): 231–244. https://doi.org/10.1002/net.3230200208
2. Kamburowski J., Michael D.J., Stallmann M.F.M. Minimizing the complexity of an activity network. *Networks*. 2000;36(1):47–52. https://doi.org/10.1002/1097-0037(200008)36:1%3C47::AID-NET5%3E3.0.CO;2-Q
3. Bianco L., Caramia M. A new formulation of the resource-unconstrained project scheduling problem with generalized precedence relations to minimize the completion time. *Networks*. 2010;56(4):263–271. https://doi.org/10.1002/net.20388
4. Brennan M. *PERT and CPM: a selected bibliography.* Monticello, Ill.: Council of Planning Librarians, 1968. 11 p.
5. Alagheband A., Soukhakian M.A. An efficient algorithm for calculating the exact overall time distribution function of a project with uncertain task durations. *Indian Journal of Science and Technology*. 2012;5(9):3310–3316. https://doi.org/10.17485/ijst/2012/v5i9.20
6. Damci A., Polat G., Akin F.D., et al. Use of float consumption rate in resource leveling of construction projects. *Front. Eng. Manag.* 2022;9:135–147. https://doi.org/10.1007/s42524-020-0118-0
7. Willis R.J. Critical path analysis and resource constrained project scheduling – Theory and practice. *Eur. J. Oper. Res*. 1985;21(2):149–155. https://doi.org/10.1016/0377-2217(85)90026-8
8. Bowers J.A. Criticality in resource constrained networks. *J. Oper. Res. Soc*. 1995;46(1):80–91. https://doi.org/10.1057/jors.1995.9
9. Shtub A. The trade-off between the net present cost of a project and the probability to complete it on schedule. *J. Oper. Manag*. 1986;6(3–4):461–470. https://doi.org/10.1016/0272-6963(86)90017-3
10. Shtub A. The integration of CPM and material management in project management. *Constr. Manag. Econ*. 1988;6(4):261–272. https://doi.org/10.1080/01446198800000023
11. Ananthanarayanan P.S. Project Technology and Management. In: S. Seetharaman (Ed.). *Treatise on Process Metallurgy*. V. 3. *Industrial Processes*. Elsevier; 2014. P. 1145–1191. https://doi.org/10.1016/B978-0-08-096988-6.00038-9
12. García-Heredia D., Molina E., Laguna M., et al. A solution method for the shared resource-constrained multi-shortest path problem. *Expert Systems with Applications*. 2021;182: 115193. https://doi.org/10.1016/j.eswa.2021.115193
13. Fulkerson D.R. Expected critical path lengths in PERT networks: *Oper. Res*. 1962;10(6):745–912. https://doi.org/10.1287/opre.10.6.808
14. Bellman R. On a routing problem. *Quart. Appl. Math.* 1958;16(1):87–90. https://doi.org/10.1090/qam/102435
15. Stern R., Goldenberg M., Saffidine A., et al. Heuristic search for one-to-many shortest path queries. *Ann. Math. Artif. Intell.* 2021;89:1175–1214. https://doi.org/10.1007/s10472-021-09775-x
16. Dreyfus S.E. An appraisal of some shortest-path algorithms. *Operations Research.* 1969;17(3):395–412. https://doi.org/10.1287/opre.17.3.395
17. Clarke S., Krikorian A., Rausan J. Computing the *N* best loopless paths in a network. *J. Soc. Indust. Appl. Math*. 1963;11(4):1096–1102. https://doi.org/10.1137/0111081
18. Pollack M. Solutions of the kth best route through a network – A review. *J. Math. Anal. Appl*. 1961;3(3): 547–559. https://doi.org/10.1016/0022-247X(61)90076-2
19. Shier D.R. Iterative methods for determining the k shortest paths in a network. *Networks*. 1976;6(3):205–229. https://doi.org/10.1002/net.3230060303
20. Shier D.R. On algorithms for finding the *k* shortest paths in a network. *Networks*. 1979;9(3):195–214. https://doi.org/10.1002/net.3230090303
21. Minieka E., Shier D.R. A note on an algebra for the *k* best routes in a network. *IMA J. Appl. Math*. 1973;11(2): 145–149. https://doi.org/10.1093/imamat/11.2.145
22. Eppstein D. Finding the *k* shortest paths. *SIAM J. Comput*. 1999;28(2):652–673. https://doi.org/10.1137/S0097539795290477
23. Yen J.Y. Finding the *K* shortest loopless paths in a network. *Manag. Sci.* 1971;17(11–Theory Series):712–716. Available from URL: http://www.jstor.org/stable/2629312
24. Hershberger J., Maxel M., Suri S. Finding the *k* shortest simple paths: A new algorithm and its implementation. *ACM Trans. Algor.* 2007;3(4): Art. 45(19 p.) https://doi.org/10.1145/1290672.1290682
25. Chen B.Y., Chen X.-W., Chen H.-P., et al. A fast algorithm for finding *K* shortest paths using generalized spur path reuse technique. *Transactions in GIS.* 2021;25(1): 516–533. https://doi.org/10.1111/tgis.12699
26. Hu X.-B., Zhang C., Zhang G.-P., et al. Finding the *k* shortest paths by ripple-spreading algorithms. *Eng. Appl. Artif. Intell*. 2020;87:Art.103229. https://doi.org/10.1016/j.engappai.2019.08.023
27. Hamed A.Y. A genetic algorithm for finding the *k* shortest paths in a network. *Egypt. Inform. J*. 2010;11(2):75–79. https://doi.org/10.1016/j.eij.2010.10.004
28. Hu X.-B., Zhou J., Li H., et al. Finding the *k* shortest paths for co-evolutionary path optimization. In: *IEEE Symposium Series on Computational Intelligence*. November 18–21, 2018; Bangalore, India. P. 1906–1912. https://doi.org/10.1109/SSCI.2018.8628928

29. Liu G., Qiu Z., Chen W. An iterative algorithm for single-pair *K* shortest paths computation. *WSEAS Transactions on Information Science and Applications*. 2015;12(Art. 30): 305–314. Available from URL: https://wseas.org/wseas/cms.action?id=10185
30. Guo J., Jia L. A new algorithm for finding the *K* shortest paths in a time-schedule network with constraints on arcs. *J. Algorithms Comput. Technol*. 2017;11(2):170–177. https://doi.org/10.1177/1748301816680470
31. Xu W., He S., Song R., et al. Finding the *K* shortest paths in a schedule-based transit network. *Comput. Oper. Res.* 2012;39(8):1812–1826. https://doi.org/10.1016/j.cor.2010.02.005
32. Jin W., Chen S., Jiang H. Finding the *K* shortest paths in a time-schedule network with constraints on arcs. *Comput. Oper. Res.* 2013;40(12):2975–2982. https://doi.org/10.1016/j.cor.2013.07.005
33. Kaufmann A., Debazeille G. *La méthode du chemin critique*. Paris: Dunod; 1964. 170 p.
34. Bellman R., Kalaba R. On *k*th best policies. *J. Soc. Ind. Appl. Math.* 1960;8(4)582–588. Available from URL: https://www.jstor.org/stable/2099049
35. Shier D.R. Computational Experience with an algorithm for finding the *k* shortest paths in a network. *J. Res. Natl. Bureau Stand*. 1974;78B(3)139–165. https://doi.org/10.6028/JRES.078B.020

About the author

Mikhail A. Anfyorov, Dr. Sci. (Eng.), Professor, Domain-Specific Information Systems Department, Institute of Cybersecurity and Digital Technologies, MIREA – Russian Technological University (78, Vernadskogo pr., Moscow, 119454 Russia). E-mail: anfyorov@inbox.ru. https://orcid.org/0000-0003-2853-6184

Об авторе

Анфёров Михаил Анисимович, д.т.н., профессор, профессор кафедры «Предметно-ориентированные информационные системы» Института кибербезопасности и цифровых технологий ФГБОУ ВО «МИРЭА – Российский технологический университет» (119454, Россия, Москва, пр-т Вернадского, д. 78). E-mail: anfyorov@inbox.ru. https://orcid.org/0000-0003-2853-6184

Translated from Russian into English by Evgenii I. Shklovskii
Edited for English language and spelling by Thomas A. Beavitt